(1)

ADVANCED DISTRIBUTED SIMULATION TECHNOLOGY

ModSAF 1.0

VERSION DESCRIPTION DOCUMENT

Ver 1.0 - 20 December 1993

CONTRACT NO. N61339-91-D-0001

D.O.: 0021

CDRL SEQUENCE NO. A007

Prepared for:

U.S. Army Simulation, Training, and Instrumentation Command (STRICOM)
12350 Research Parkway
Orlando, FL 32826-3276

3298

Prepared by:

LORAL
Systems Company

ADST Program Office
12151-A Research Parkway
Orlando, FL 32826

REPORT DOCUMENTATION PAGE

Form approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 12/20/93	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE Version Description Document - ModSAF 1.0	5. FUNDING NUMBERS C N61339-91-D-0001, Delivery Order (0021), ModSAF (CDRL A007)
6. AUTHOR(S) Joshua E. Smith, Anthony J. Courtenanche, Daniel A. Cifin	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Loral Systems Company ADST Program Office 11251-A Research Parkway Orlando, FL 32826	8. PERFORMING ORGANIZATION REPORT NUMBER ADST-TR-W003269
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Simulation Training and Instrumentation Command (STRICOM) 12350 Research Parkway Orlando, FL 32826-3275	10. SPONSORING ORGANIZATION REPORT ADST-TR-W003269
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE A

13. ABSTRACT (Maximum 200 words)

This document provides instructions for installing, building, operating, and porting the ModSAF 1.0 application.

14. SUBJECT TERMS Modular Semi-Automated Forces, DIS, ADST, BDS-D	15. NUMBER OF PAGES Approx 40
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	17. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	17. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Table of Contents

1 Release Notes 1

1.1 Distribution 1

1.1.1 tools 2

1.1.2 libsrc 2

1.1.3 include 3

1.1.4 data 3

1.1.5 lib 3

1.1.6 info 4

1.1.7 src 5

1.1.8 bin 6

1.1.9 terrain 6

1.1.10 profiles 7

1.1.11 scenarios 7

1.2 Executable Version of ModSAF 1.0 8

1.3 Building ModSAF 1.0 9

1.4 Running ModSAF 1.0 14

1.4.1 Command Line Arguments 15

1.4.1.1 SAFStation 15

1.4.1.2 SAFSim 15

1.4.1.3 Networking 16

1.4.1.4 Terrain 17

1.4.1.5 Site Customization 17

1.4.2 Configuring ModSAF 18

1.4.3 Known Problems 18

1.5 Porting ModSAF 1.0 21

1.6 Printing ModSAF Documentation 23

1.7 Troubleshooting 24

Index 27

1 Release Notes

The sections in this chapter provide information about ModSAF distribution and documentation. They also provide instructions for installing, building, operating, and porting the ModSAF application.

1.1 Distribution

There are two ModSAF 1.0 software distributions. The source code distribution contains all the source files needed to build the ModSAF program, and all the data file needed to run. Executables must be compiled after software installation. The executable distribution contains all of the executable and data files needed to run the ModSAF program.

The distribution includes 150 libraries and a source directory which break down as follows:

```
Total lines = 326335

        Whitespace = 43336

        Comments = 67393

        Inline = 5062

        C = 210544

        Source (C + Inline) = 215606

Percentage whitespace:  13.28 %
Percentage comments:    20.65 %
Percentage C:           66.07 %

Total lines in data files = 27808

Total lines of TeXinfo documentation = 103256
```

The root of the distribution directory structure is called 'common/'. This directory contains the subdirectories which are described in the following sections.

1.1.1 tools

Each '`Makefile`' in the ModSAF source tree references a common set of compiler configuration files. These files specify which compiler to use, what flags to specify, and the steps required to compile and install software archives, executables, documentation, and data files.

The '`tools`' directory contains the compiler configuration files. Included in the configuration files is the following:

- Configuration for Mips Magnum or M/2000 systems, using RISC/OS 4.50, 4.51, or 4.52. cc version 2.11 is required.
- Configuration for Silicon Graphics hardware and operating system.
- Configuration for Sun 4 systems, using the Gnu gcc compiler. The native Sun compiler can probably also be used, with some modification of the files in this directory (see Section 1.5 [Porting ModSAF 1.0], page 21).
- Prototype configuration for IBM AIX systems.

The Mips, SGI and SUN configurations are officially supported. Portions of the ModSAF software have been compiled and run on IBM, and HP systems, but the full system has not been built or tested. The tools directory is not included in the executable version of ModSAF 1.0.

1.1.2 libsrc

The directory '`common/libsrc`' contains the sources for all the software libraries which make up the ModSAF system. The ModSAF architecture is built up from many libraries (150 in the ModSAF 1.0 distribution). Most of these (142) are considered *ModSAF compliant*, which means they meet a set of coding standards which are spelled out in another document (see section 'Coding Standards' in Software Architecture Design and Overview Document). The remaining libraries (8) are legacy from the SIMNET project, and have been used without modification in ModSAF. During the initial build process, the file '`common/libsrc/modsaf.libs`' is created which contains a list of the ModSAF compliant libraries. None of the '`common/libsrc`' libraries are included in the executable version of ModSAF 1.0.

Overviews of the functionality of these libraries can be found in the topmost ModSAF `info` node, `modsafdir`.

1.1.3 include

The directory '`common/include`' contains two source subdirectories:

'`common/include/global`'
: This directory contains globally referenced C header files, which contain definitions of physical constants, conversion factors, dynamic memory allocation conventions, and type definitions.

'`common/include/protocol`'
: This directory contains the definition of the DIS, SIMNET, and PO protocols, as well as definitions of many experimental protocols which have been used in SIMNET.

In addition, there is the directory '`common/include/libinc`' where the C header files defined by the various libraries are deposited as part of the compilation process.

The directory '`common/include`' is not included in the executable version of ModSAF 1.0.

1.1.4 data

The directory '`common/data`' acts as a repository for data files defined in the ModSAF libraries. Data files are copied from library directories into this directory during the compilation process. The files in this directory can be identified by their extensions:

'`.rdr`'	Data files formated using the '`libreader`' format. These data files are used by the simulation and workstation applications at run time.
'`.xrdb`'	X windows resource definition database files. These data files are incorporated into the X Server Resource Database at, or just prior to, run time.
'`.map`'	A binary format used by '`libhm`'.

1.1.5 lib

The directory '`common/lib`' contains a subdirectory structure which acts as a repository for compiled software library archives (`ar(1)`). The directory structure is constructed automatically as part of the compliation process. First, there are subdirectories for each architecture which is built. The architecture is specified via the environment variable `BUILD_ARCH` and should have one of the values: **sgi**, **mips**, **sun4**, or **aix**. If you use csh or tcsh, this can be set in the '`.cshrc`' file in your home directory using the command:

```
setenv BUILD_ARCH sgi
```

(Substitute your own architecture for **sgi**.) If you use sh or ksh, this can be set in the '`.profile`' file in your home directory using the command:

```
BUILD_ARCH=sgi
export BULD_ARCH
```

The '`.cshrc`' modifications are made automatically upon execution of the '`make-all`' script (see Section 1.3 [Building ModSAF 1.0], page 9).

Below the architecture directory, you will find a subdirectory which is specific to the application configuration which is built. For ModSAF, this will always be called '`modsaf`'. This name comes from a configuration file, which is also indicated by an environment variable. For csh or tcsh users, this should be set:

```
setenv BUILD_APP src/ModSAF/modsaf.config
```

For sh or ksh:

```
BUILD_APP=src/ModSAF/modsaf.config
export BUILD_APP
```

As with the **BUILD_ARCH** variable, this is automatically place in your '`.cshrc`' by '`make-all`'.

The files in the application subdirectories are merged into the ModSAF executable, by the linker (`ld(1)`). This directory is not included in the executable version of ModSAF 1.0.

1.1.6 info

The directory '`common/info`' contains the ModSAF documentation in Emacs-info format. The files in this directory are compiled from '`.texinfo`' files in the library and source directories, and are installed during the compilation process.

1.1.7 src

The directory '**common/src**' contains sources for building executable programs and operating system related modules:

'**common/src/ModSAF**'
: This directory contains the following:

  - The ModSAF executable program and compilation script.
  - The '**common/src/ModSAF/entities**' subdirectory which contains vehicle parameter files.
  - The '**common/src/ModSAF/docs**' subdirectory which contains the top-level ModSAF 1.0 documentation.

'**common/src/interck**'
: The source and executables for the interck bug finder program reside in this directory. This directory is not included in the executable version of ModSAF 1.0.

'**common/src/blaster**'
: The source and executables for the network blaster program reside in this directory. This directory is not included in the executable version of ModSAF 1.0.

'**common/src/OSAtemplate**'
: This directory contains the template files used by the **osatemplate** script found in the bin directory. This directory is not included in the executable version of ModSAF 1.0.

'**common/src/logger**'
: The logger source files and executable program reside in this directory.

'**common/src/cmc**'
'**common/src/simle**'
: These directories contain C header files and compiled operating system modules. These directories are not included in the executable version of ModSAF 1.0.

'**common/src/depends**'
: **depends** is a program which examines ModSAF compliant software directories and identifies the dependencies between them. It is used to do ordered compilation, build Makefiles, and generate FIG format dependency graphs. This directory is not included in the executable version of ModSAF 1.0.

'**common/src/public**'
: The '**common/src/public**' directory contains public domain source files for **xinfo**, **makeinfo**, and **make-3.68**.

  - **xinfo** is a public domain X windows interface tool for reading Emacs-info format documentation without using emacs. It is included as a convenience for reading the ModSAF documentation.

- **makeinfo** is a public domain tool for creating Emacs-info format documentation from our texinfo documentation.
- **make-3.68**, also known as **gmake**, is a public domain make tool used in building the ModSAF application.

The compiled versions can be found in the 'common/bin' directories. These sources are provided as a convenience to the ModSAF customer and are not supported. This directory is not included in the executable version of ModSAF 1.0.

1.1.8 bin

The directory 'common/bin' contains subdirectories for each of the supported architectures: **sgi**, **mips**, **sun4**, **aix**. As explained earlier (see Section 1.1.5 [lib], page 3), this is specified by the environment variable BUILD_ARCH. Since this directory contains executable programs which are needed to compile ModSAF, it should be added to your program search PATH. For csh or tcsh users, this is done by adding it to the '.cshrc' file. For example:

```
set path = (/usr/staff/myname/common/bin/sgi . /bin /usr/bin /usr/bin/X11)
```

For sh or ksh users, the syntax is:

```
PATH=/usr/staff/myname/common/bin/sgi::/bin:/usr/bin:/usr/bin/X11
export PATH
```

For csh or tcsh users, the 'make-all' script will place the correct directory in your path.

The 'common/bin/<BUILD_ARCH>' subdirectory contains an awk script 'fsm2ch.awk', which is used during the compilation of ModSAF task libraries. The 'fsm2ch.awk' file is originally located in 'common/libsrc/libtask' directory and is installed in the correct 'common/bin' subdirectory upon running the **make-all** script. The directories in 'common/bin' also act as a repository for compiled executables, such as the 'depends' program, 'makeinfo', 'gmake', and 'xinfo'. The executables for 'makeinfo', 'gmake', and 'xinfo' along with the 'osatemplate' script are distributed in the correct 'common/bin' subdirectory.

1.1.9 terrain

The directory 'common/terrain' contains a subdirectory for each terrain database. The distribution includes the following databases:

'**common/terrain/hunter-0110**'
: A version of the 50KM x 50KM Hunter-Liggett terrain database.

'**common/terrain/itsec93-0101**'
: A 100KM x 100KM superset of the Hunter-Liggett terrain database for use at the I/ITSEC '93 show.

'**common/terrain/ntc-0101**'
: A version of the 50KM X 50KM National Training Center terrain database.

'**common/terrain/knox-0311**'
: A version of the 50KM x 75KM Ft. Knox, Kentucky terrain database.

1.1.10 profiles

The directory '**common/profiles**' is created when the ModSAF program is first run, to hold user-customization profile files.

1.1.11 scenarios

The directory '**common/scenarios**' is created when the ModSAF program is first run, to hold user-created scenario files.

1.2 Executable Version of ModSAF 1.0

The directories included in the executable version of ModSAF 1.0 are:

```
common/data
common/info
common/terrain
common/terrain/hunter-0110
common/terrain/ntc-0101
common/terrain/knox-0311
common/src
common/src/logger
common/src/ModSAF
common/scenarios
common/profiles
common/overlays
common/logs
```

The following steps should be followed to load the ModSAF 1.0 program from the distribution tape:

Getting Started

Choose a directory location on your machine for the executable code to reside in, such as '`/usr/staff/myname/modsaf`'. *Do not install the ModSAF 1.0 distribution "over" a previous ModSAF release.* Make sure you are in this directory before you begin the next step.

Unloading the tape requires approximately 50 Mbytes of disk storage for the executables and the terrain databases.

Load the software

Load ModSAF 1.0 Tape in tape drive. If you are on an SGI machine, unload the tape as follows:

for builtin drive:

```
tar xvf /dev/tapens
```

for external SCSI bus tape drive (replace # with SCSI device number)

```
tar xvf /dev/rmt/tps0d#ns
```

If you are on a Mips machine, unload the tape as follows:

```
tar xvf /dev/rmt/ctape0
```

The name of the tape device differs on other machines. Once the executables are installed skip ahead to the running instructions (see Section 1.4 [Running ModSAF 1.0], page 14).

1.3 Building ModSAF 1.0

The following steps should be followed to build the ModSAF 1.0 program:

Getting Started

Choose a directory location on your machine for the source code to reside in, such as '`/usr/staff/myname/modsaf`'. *Do not install the ModSAF 1.0 distribution "over" a previous ModSAF release.* Make sure you are in this directory before you begin the next step.

Unloading the tape requires approximately 80 Mbytes of disk storage for source code, executables, documentation, and the terrain databases. Once fully compiled, approximately 170 Mbytes of disk storage is consumed by the source code, object code, terrain, and documentation.

All the executable programs ('`xinfo`', '`gmake`', '`osatemplate`', '`makeinfo`') which are needed to build ModSAF are on this tape.

The build process assumes the machine has a C compiler.

Load the software

Load ModSAF 1.0 Tape in tape drive. If you are on an SGI machine, unload the tape as follows:

for builtin drive:

```
tar xvf /dev/tapens
```

for external SCSI bus tape drive (replace # with SCSI device number)

```
tar xvf /dev/rmt/tps0d#ns
```

If you are on a Mips machine, unload the tape as follows:

```
tar xvf /dev/rmt/ctape0
```

The name of the tape device differs on other machines.

This will load all the ModSAF software, as described in the previous section. On an SGI machine, the system may ask you to supply a second tape. Ignore this request by hitting return.

Prepare your environment

The compilation of ModSAF requires that a few environment variables be set (see Section 1.1.5 [lib], page 3). The values you should use for these environment variables depends upon the architecture you will be running on and the way you will be using the ModSAF program. The ModSAF program can be built many different ways within the *same* source hierarchy. For example, executable versions of ModSAF for both a Sun 4 and an SGI can be built from the same set of sources.

The architecture is set via the environment variable `BUILD_ARCH`.

Most ModSAF 1.0 users should build the application using the standard application configuration, described by a file in the application source directory '`src/ModSAF/modsaf.config`'. This file specifies the set of compilation flags which select which features are enabled and which are not. There is another configuration, called '`src/ModSAF/safsim.config`' which builds the application without X windows.

The application is set via the environment variable `BUILD_APP`.

Many programs used in the compilation of ModSAF are installed in the directory '`common/bin/<BUILD_ARCH>`'. This must be in your `PATH` for compilation to succeed (see Section 1.1.8 [bin], page 6).

The ModSAF program is very large, and many compilers will need extra disk space to build it. Most architectures support an environment variable called `TMPDIR` which specifies a directory in a large disk partition. You will probably need to specify such a directory. For example:

```
setenv TMPDIR /usr/tmp
```

Finally, you must select a set of `EXTRA_CFLAGS` to suit your needs. This environment variable specifies extra flags which are passed to the C compiler when the program is built:

'`-g`' This specifies that a symbol table should be included, so that the program can be debugged using '`dbx`'.

'`-O`' This specifies that the program should be optimized to improve execution speed. The exact version of this flag differs between machines, and some machines do not allow this flag together with -g. On an SGI or a Mips, the flags '`-g3 -O2`' will yield the maximum optimization which current compiler versions can perform on ModSAF. *WARNING: Optimized code generally cannot be debugged reliably using dbx.*

'`-mips2`' Newer versions of SGI hardware and compilers support the MIPS II instruction set. If you have such a machine, the '`-mips2`' option will yield up to a 2x performance boost, with no negative effects. This argument *does not* interfere with debugging.

See the Unix manual page on '`cc`' for more information. If multiple arguments are used, they must be combined with quotations. For example:

```
setenv EXTRA_CFLAGS '-g3 -O2'
```

If you do not set these environment variables yourself, the '`make-all`' script will set up a standard configuration in the '`.cshrc`' file in your home directory.

Compile the ModSAF software

To compile the ModSAF software:

```
cd src/ModSAF
```

`make-all`

If necessary, the '**`make-all`**' script will set a few necessary environment variables and put the correct '`common/bin`' subdirectory in your path before building anything (see Section 1.1.5 [lib], page 3).

'**`make-all`**' makes the '**`depends`**' program, many software libraries, and the ModSAF application, which is called '**`modsaf_`***arch*'. Where *arch* is replaced with the architecture you are building on. For example, the executable would be called '**`modsaf_sgi`**' for the Silicon Graphics architecture.

You will get an error message in certain libraries which do not have TeXinfo documentation, as the build process attempts to generate Info documentation in every library. These error messages can be safely ignored.

When the build is complete, the modsaf application will be test run to have it display its arguments and verify that the executable does run.

Install the simle driver (Sun only)

If you are running ModSAF on a SPARC based SunOS 4.1.1 (or higher), you can use the installable simle driver to send and receive SIMNET libassoc packets. There are several files used to load the simle driver, all of which are in the directory common/src/simle/dvr:

'**`simle_rc`**'
: Contains the lines that should be put into the file /etc/rc.local to load the driver at boot time.

'**`simle_load`**'
: A script to do the loading, which should live in /etc.

'**`simle_mknod`**'
: A script to make the necessary "/dev/simle*" file, which should live in /etc.

'**`simle.o`**'
: The driver object, which should live in /etc.

'**`simle_install`**'
: A script to put the above 3 files in /etc, add the contents of simle_rc to /etc/rc.local (if not already there), execute /etc/simle_load, and configure up to 2 ethernet cards that may be present. This change to rc.local is permanent and means the simle driver will be loaded into the kernel each time the machine is rebooted. You must be root to run this script.

To install this driver, run the **`simle_install`** script as root.

Use the xinfo documentation tool

Determine the absolute info path where the info sources have been installed. If you installed the software in '`/usr/staff/myname/modsaf`', the *<absolute-info-path>* will be '`/usr/staff/myname/modsaf/common/info`'.

```
cd /usr/staff/myname/modsaf
common/bin/xinfo -file modsafdir -path <absolute-info-path>
```

You may click on various text nodes to follow the ModSAF documentation tree. When finished, click "quit" to exit.

Customize emacs to use info documentation:

If you are unfamiliar with the use of Info in emacs, you can enter emacs and type `^h i`. Read the instructions and type **h** to get a tutorial on the use of Info.

To access modsaf info documentation from emacs, you will have to edit the `dir` info node to access the `modsafdir` info node:

1. Enter emacs. To find the filesystem location of the `dir` node,
2. Type `^h i` This will bring you into the info system at the `dir` node.
3. Type `^x^b` to get a buffer list. The absolute pathname of the `dir` node will be listed.
4. Exit emacs. As root, you can edit the '`dir`' file in the directory identified in the previous step to add the following line to the end of the file:

   ```
   * ModSAF: (<absolute-info-path>/modsafdir).
             Documentation about ModSAF.
   ```

   Once the `dir` node is modified, emacs will be able to access the modsaf info sources directly:

   1. Enter emacs
   2. Type `^h i`
   3. Type `m`, followed by `modsaf` to enter the modsaf node.

Create /etc/assoc.def

Create the file '`/etc/assoc.def`' file which will describe the simulation site and host of the machine which will run the ModSAF application. Site numbers are typically assigned to a site by Loral. Host numbers are assigned by the site manager at a given site. As an example, if you are site 1 and the machine you are on is host 5, you can create the '`assoc.def`' file as follows (as root):

```
cat > /etc/assoc.def
site 1
host 5
^D
```

Configure the real time clock (SGI only)

The SGI operating system defaults to an extremely low clock resolution of 10 ms. This can be fixed using the command '`/etc/ftimer -f on`'. This command requires *root* privileges. You can set up the machine to automatically fix the real time clock by creating a boot-up initialization script:

```
cat > /etc/rc2.d/S99ftimer
/etc/ftimer -f on
/etc/ftimer
^D
chmod +x /etc/rc2.d/S99ftimer
```

1.4 Running ModSAF 1.0

If you are running on an SGI system, and you want to use the network, you must login as root. Remeber that as root, any command you issue from the current directory must begin ./<command>. Once the ModSAF program has been successfully compiled, it can be run from the directory '**common/src/ModSAF**'. There are actually two ModSAF programs, which are combined into a single executable. For the Mips architecture, the executable file is called '**modsaf_mips**'; for the SGI, '**modsaf_sgi**', and so on. The two ModSAF programs are:

SAFStation
: This is the ModSAF user interface. This program provides a map display, and allows a user to create units, missions, and make assignments to ModSAF vehicles.

SAFSim
: This is the ModSAF vehicle simulation program. This program creates SAF vehicles in response to requests from the SAFStation program (or other programs running on the network), and controls the behavior of those units.

For testing, these two programs can be run together on a single computer. However, such a configuration may not be able to meet real-time simulation requirements because of the burden of user interface processing.

The ModSAF user interface uses the X resource manager for selection of layout, sizing, coloring, etc. The resources used by ModSAF can be found in the file '**common/src/ModSAF/ModSAF**'. If you are using a machine with X windows version X11R5, you can set up the system to automatically load this file when the program starts by adding the following to your '**.cshrc**':

```
setenv XAPPLRESDIR .
```

or if you use a variant of sh, add this to your '**.profile**':

```
XAPPLRESDIR=.
export XAPPLRESDIR
```

If, however, you are using X windows version X11R4, a bug in the X resource manager prevents this from working. On such machines, you must use the xrdb command every time you login. The executable for xrdb will be found in your X11 bin directory, either /usr/X11/bin, or /usr/local/X11/bin. This directory should replaced <Xbindir> in the command below:

```
cd /usr/staff/myname/modsaf/common/src/ModSAF
/<Xbindir>/xrdb -merge ModSAF
```

The above two commands need to be done anytime you login, after logging in, to configure the X server to use the default resources.

1.4.1 Command Line Arguments

The ModSAF program takes many command-line arguments, most of which are described in the following sections. The default values for these arguments are compiled into the ModSAF program, but can be overridden with an environment variable. The name of the environment variable depends on your architecture. The Silicon Graphics Architecture uses `MODSAFSGIARGS`, the Mips uses `MODSAFMIPSARGS`, and the SUN architecture uses `MODSAFSUN4ARGS`. For example, to change the default terrain database to `knox-0311` on an SGI, add the following to your '`.cshrc`':

```
setenv MODSAFSGIARGS '-terrain knox-0311'
```

or if you use a variant of '`sh`', add this to your '`.profile`':

```
MODSAFSGIARGS='-terrain knox-0311'
export MODSAFSGIARGS
```

For an overview of the command line arguments and to see their default values, use (the program name varies):

```
cd common/src/ModSAF
modsaf_sgi -help
```

1.4.1.1 SAFStation

The command line argument `-gui` selects the SAFStation functionality. To disable the SAFStation, instead use `-nogui`.

In addition, all standard X windows command line options are supported. For example, to set the display device, use `-display` *machine*`:0`.

1.4.1.2 SAFSim

The command line argument `-simulate` selects the SAFSim functionality. To disable the SAF-

Sim, instead use **-nosim**.

Another command line argument which impact the simulation is the simulation loading factor **-load** *floating point number*. This option specifies the portion of maximum simulation load imposed by one simulated vehicle, and is used in the inter-machine load balancing algorithm. Since a typical workstation can support about 60 local vehicles, the value 1/60 (0.167) is used.

1.4.1.3 Networking

The ModSAF program can be configured to support many different networking schemes. To select a networked ModSAF, use the option **-network**. To disable networking use **-nonet**.

The network device can be selected with the argument **-netdev** *device*. The appropriate network device varies on different machines.

The DIS and SIMNET protocols support multiple simultaneous *exercises* running on a single network. To specify the exercise in which a ModSAF should participate, use the command line option **-exercise** *1-254*.

The ModSAF system uses the Persistent Object protocol for communications between SAFSims and SAFStations. The protocol allows multiple simultaneous persistent object databases within a single exercise. All ModSAF computers which are using the same database will work together to do SAF simulations. To specify the database being used by a set of ModSAF simulators, use the command line option **-database** *1-254*.

Simulation protocols must use an application layer protocol to move packets between computers. In SIMNET this function was performed by the *Association Layer Protocol* (libassoc). In DIS, for the time being, this function is being performed with the Internet *User Datagram Protocol* (UDP/IP). The ModSAF software allows either protocol to be used, regardless of whether SIMNET or DIS simulation protocols are being used. To use the Association Layer Protocol, use **-assoc -noudp**. To use the User Datagram Protocol, use **-udp -noassoc**.

Note that the Association Layer Protocol requires a modified operating system kernel on the Mips Magnum, and extra hardware and a modified kernel on the Mips M/2000.

Using the Association Layer Protocol on a Sun requires some extra configuration (see Section 1.3 [Building ModSAF 1.0], page 9).

The UDP networking software can run either asynchronously (driven by software signals generated by the operating system), or via polling. The performance characteristics of each method will differ between different operating systems. To select asynchronous operation use **-asynch**; for polling, use **-synch**.

No standard exists for which UDP port should be used for transmitting different protocols (although 3000 is most common for DIS). To select the port for DIS protocol, use `-disport` *port*. To select the port for PO protocol, use `-poport` *port* `-noexperimental`. To select the port for Stealth protocol, use `-stealthport` *port*.

Another option for transmitting the PO protocol under UDP/IP is to package all PO packets into a single experimental DIS PDU kind. To do this, use `-experimental` *kind (134-255)*.

The ModSAF program supports both SIMNET 6.6.1 and DIS protocols. To use DIS, use `-dis`. For SIMNET, use `-simnet`.

The current defacto standard DIS version is that used at the I/ITSEC '93 interoperability demonstration (DIS 2.0.3). Although this version is the default, it can be selected by using the `-version 3` option. To select the ITSEC '92 version (DIS 1.0), use `-version 1`. Note that these two versions are not compatible. Every computer on the network must be using the same version of the DIS protocol for interoperation. DIS version 2.0.2 (-version 2) is no longer supported.

1.4.1.4 Terrain

The terrain database can be selected with the command line option `-terrain` *name*. The *name* can be the name of a terrain database in the directory '`common/terrain`', or it can be the absolute pathname of a terrain database directory (starting with '`/`').

1.4.1.5 Site Customization

The simulation address for a machine should be set in the file '`/etc/assoc.def`'. This address can be overridden with the command line option `-simaddr` *site (1-65535) host (1-65535)*.

The ModSAF user can create user interface profiles (for example, the default coordinate system, metric vs English, etc.), and save them to disk. The default directory for these files is '`common/profiles`', however this can be overridden with `-profile` *directory*.

The ModSAF user can save scenarios (created units and command graphics) to disk. The default directory for these files is '`common/scenarios`', however this can be overridden with `-scenario` *directory*.

Some features of the ModSAF user interface require *privileges* to access. These privileges, are password protected, by default. The default password, for historical reasons, is '`foozball`'. This password can be modified by creating a file in the '`common/src/ModSAF`' directory called '`.password`'. For example, to change the password to '`frabnitz`', type:

```
cd common/src/ModSAF
cat > .password
frabnitz
^D
```

To disable password protection, create an empty '`.password`' file:

```
cd common/src/ModSAF
rm -f .password
touch .password
```

1.4.2 Configuring ModSAF

By default, when the ModSAF program is run from the directory '`common/src/ModSAF`', data files ending in '`.rdr`' are loaded from the directory '`../../data`', which translates to '`common/data`'. However, these data files can be overridden by placing modified versions in the directory from which the program is run. Instead of loading the default files, the modified versions will be loaded.

The default X resources used by ModSAF can be overridden using the X resource manager. Typically, this is done by placing resource modifications in a user's '`.xresources`' file. The exact name of a widget resource can be found by looking in the '`.xrdb`' file which configures the widget by default. Also, some possible X resource customizations are described in the ModSAF library TeXinfo documentation.

1.4.3 Known Problems

The following is a list of known problems with the ModSAF 1.0 release.

- The execution matrix is an experimental piece of software which has several known problems:

  - A mission for a unit cannot be saved until after it is assigned.
  - The ability to edit an execution matrix after it has been assigned is extremely limited. Many editing operations will have undesired effects.
  - There is no ability to specify global mission parameters such as fire permissions, or enemy situation. These parameters must be specified for every frame.
- The logger has the following list of problems
  - The logger does not yet support The DIS 2.0.3 protocol. It only supports Simnet and DIS 1.0.
  - Recording over part of an existing file does not work.
  - The Internal button doesn't work unless the Network button is also activated
- In some circumstances, M2, BMP1, and BMP2 will attempt and fail to engage targets with missiles when the target is not in the same plane/orientation as the attacker.
- Due to the use of absolute elapsed time instead of relative time, the task state for VEnemy changes more frequently than necessary.
- Turret scan sectors are often incorrect during road marches.
- Parameter changes made to a platoon or company after creation are not propogated down to the individual vehicles. For example, you cannot teleport a platoon to a new location by editing the platoon.
- When a vehicle is shooting at an enemy and runs out of that type of munition it will not shoot even if it is assigned more ammunition.
- There are a few munitions that are currently not defined in the vehicle's vassess parameter file. This means the vehicle can not shoot that munition.
- Many soviet formations are not correct. One effect of this is that vehicles will cross paths when occupying a position.
- The "Conform to Terrain" option in the Move task frames is an experimental piece of software, which often leads to incorrect movement.
- Occasionally an M2 will not start moving again after its stops to fire its TOW missile.
- Some mixed platoons don't have the right force id (friendly/enemy).
- You can't stop a air to air intercept with the STOP REACTION button.
- Tanks cannot shoot DI yet. (US M59, USSR D) They have a weapon for it, but it is not listed in the vehicle's VAssess section of the rdr file.
- Selecting a stealth from the map display when the Unit Operations menu is present will lead to overlapping control menus. Exit the Unit Operations menu before choosing a stealth.
- After executing several hard turns, the fixed wing aircrafts may lose altitude and may not be able to recover.
- Fixed wing aircraft can not perform the task "Fly Route". If assigned this task, they will just sit there. FWA should only be assigned "Sweeep", "Cap", and "Return to Base".

- Rotary wing do not currently have a landing task. They can only be told to return to base.
- Rotary wing aircraft performing takeoff right now will either spin a bit, or bump the ground. They will then perform normally.
- Rotary wing aircraft performing a fly route will occasionally in pasing a route point turn around and go back to it, then continue on their route.

1.5 Porting ModSAF 1.0

The ModSAF software is quite portable. It uses K&R C, with modest extensions (such as unique identifiers which exceed 8 characters).

X windows is used exclusively for windowing (SGI GL is *not* used). X versions 4 and 5 both work well with ModSAF (R5 seems to have improved performance on the Mips platform). The OSF/Motif 1.1 widget set is used.

The first step in porting ModSAF to a new platform is to edit the configuration in the '`common/tools`' directory. Most of the configuration parameters are in the file '`make.config`'. This file include specification of the C compiler and options.

You must give a name to the new architecture, for example, if you are porting to a Hewlett Packard operating syste, you could call the architecture '`hp`'.

The application description file (such as '`common/src/ModSAF/modsaf.config`') must include this new architecture name in the list of supported architectures (`ARCHLIST`).

Edit the file '`common/tools/make.config`' to include any special compiler, linker, archiver, etc. directives. In the middle of this file you will find a set of flag settings for each architecture. Copy an existing architecture and modify it as needed. This may include some of the following:

One common change is for machines which do not have the single-precision square root function, `fsqrt`. On such machines add the definition `-Dfsqrt=sqrt` to the `<arch>_CFLAGS`.

Another difference between machines is whether they have or need the command '`ranlib`' executed on library archives. This is selected by the `<arch>_RL` and `<arch>_RLFLAGS`.

It is unlikely that a new machine will be able to easily support the SIMNET Association Layer Protocol, because it requires access to raw 802.3 Ethernet packets. On such machines, skip the compilation of '`libnetif`', '`libassoc`', and '`libp2p`' and add `-DNO_ASSOC` to the `<arch>_CFLAGS`.

Another common difference between unix systems is whether the string operation header file is called '`string.h`' or '`strings.h`'. If it is the former, then add the definition `-DUSESTRINGDOTH` to the `<arch_>CFLAGS`.

Once the tools directory is set up for a machine, most directories should compile without modification.

1.6 Printing ModSAF Documentation

The complete ModSAF documentation set is roughly 1000 pages, so it makes sense to only print documents selectively. The documents are written using the TeXinfo documentation system. Printing the documents requires access to the TEX software package. TEX is available from the University of Washington, for a small fee.

You will need the TEX input file '`texinfo.tex`'. If you do not have this file, one has been provided in '`common/bin`'. Copy this to the directory where TEX looks for input files (often in a directory called '`inputs/`', or '`macros/`').

There are rules in each '`Makefile`' which can translate the '`.texinfo`' documents into '`.dvi`' documents, the output of TEX. For example, to generate the documentation for '`libctdb`', use:

```
cd common/libsrc/libctdb
make libctdb.dvi
```

'`.dvi`' files must be translated to a printer's native language before they can be printed. One tool commonly used for this translation is '`dvi2ps`'. Several conversion tools are included in the TEX distribution.

1.7 Troubleshooting

When I compile an individual library, I get lots of error messages

The ModSAF Makefiles now expect to use the GNU make program, gmake. Did you compile with '`gmake`'?

Are the `BUILD_ARCH` and `BUILD_APP` environment variables set correctly?

Is the correct '`bin`' subdirectory in your path?

See Section 1.3 [Building ModSAF 1.0], page 9.

See Section 1.1.5 [lib], page 3.

See Section 1.1.8 [bin], page 6.

When I start the program with `-gui` *I get an X error.*

The user interface is all mashed up into the upper left screen corner.

If you are using X11 R4, did you run '`xrdb -merge ModSAF`'?

If you are using X11 R5, did you '`setenv XAPPLRESDIR .`'?

See Section 1.4 [Running ModSAF 1.0], page 14.

When I start the program under '`dbx`' *I get an I/O error.*

The asynchronous UDP implementation uses the SIGIO signal to receive incoming packets. '`dbx`' can be told to ignore this signal by adding the command:

```
ignore 23
```

to the file '`.dbxinit`' in your home directory.

I get a warning from '`cpp`' *about redefinition of M_PI.*

I get a warning from '`cpp`' *about redefinition of NULL.*

These warnings have been seen on the SGI, and we have not yet identified where the redefinition is occurring. They can be safely ignored.

My C compiler dies with a "symbol table full" error.

The ModSAF program is very large, and has been known to choke compilers. Check your operating system reference manuals to see if the compilers limits can be increased.

My C compiler dies with an internal error while linking.

Some C compilers use the directory '`/tmp`' for storing internal files during compilation. There may not be enough space in the root partition to hold some of these files. If this is the case, the compiler can often be told to redirect temporary files to a different directory (the Mips uses the environment variable `TMPDIR`).

I can't lint the ModSAF program.

Although there is a lint target in each library, many of the libraries include many X and Motif header files, which can overwhelm lint. The **interck** program finds many of the same errors lint can find, and many more.

'`dbx`' *shows incredible values for local variables.*

'`dbx`' *complains that the program has no symbol table.*

By default, the ModSAF program is set up to compile with optimization. This makes use within a debugger difficult or impossible. Change the definition of your environment variable `EXTRA_CFLAGS` to include a symbol table (usually '`-g`') and omit optimization. See Section 1.3 [Building ModSAF 1.0], page 9.

Network performance at my site degrades badly when I run ModSAF.

The DIS protocols currently use *broadcast* UDP/IP for packet transmission. These packets will be received by every computer on the network, and some may respond with "I hear you but I don't know why you are telling me" packets (Mips RISC/OS does this). The SIMNET Association Layer Protocol, in contrast, is *multicast* so that only those computers which explicitly ask for the packets will hear them. The ModSAF program allows the DIS protocol to be sent using the SIMNET Association Layer Protocol (as protocol family 3) with the command line arguments `-assoc -noudp -dis`.

See Section 1.4.1.3 [Networking], page 16.

I need to run in an area for which no terrain database is available.

The library '`common/libsrc/libctdb`' contains a program called '`ocean`', which can be used to create nominal (flat) databases anywhere in the world. Go to that directory, '`make ocean`', and run '`ocean -help`' for details.

How can individuals use different compiler configurations?

The environment variables `EXTRA_CFLAGS` and `EXTRA_LDFLAGS` can be set by different users to augment the compilation process. These flags are used *in addition to* those specified in '`common/tools/make.config`'.

ModSAF says it is out of cycles.

This is a normal error which indicates that the simulation capabilities of the ModSAF program have been exceeded. Performance can be improved on some architectures by using optimization and special compiler directives (such as -mips2 on the SGI). Also, SGI users may encounter this error if the fast ftimer is not turned on (See Section 1.3 [Building ModSAF 1.0], page 9).

Index

A

AIX 2
assoc.def 12

B

blaster 5
BUILD˙APP 3, 9
BUILD˙ARCH 3, 9

C

common/bin 6
common/data 3
common/include 3
common/info 4
common/lib 3
common/libsrc 2
common/profiles 7
common/scenarios 7
common/src 5
common/terrain 6
common/tools 2
Compiling 10
Configuration 2
Constants 3

D

Data Files 3, 18
Debugging 9
depends 5, 6
Disk Space 8, 9
dvi 23

E

Ethernet 802.3 21
EXTRA˙CFLAGS 9

F

fsm2ch.awk 6
fsqrt 21
ftimer 9, 12

G

gmake 6

H

Header Files 3
Host 12

I

IBM 2
Info 4, 12, 23
interck 5

K

K&R C 21

L

libassoc 21
libhm 3
libnetif 21
libp2p 21
libreader 3

M

Macros 3
make-all 3, 9, 10
makeinfo 6
Mips 2
ModSAF Compliance 2
modsaf.libs 2

N

Networking 5

O

Operating Systems 5
Optimization 9
osatemplate 5

P

PATH 9
Printing 23

Protocols 3

R

ranlib 21

S

SGI GL 21
Silicon Graphics 2
simle 11
Site 12
string.h 21
Sun 2, 11

T

tar 8, 9
Terrain Databases 6
TeX 23
TeXinfo 4, 23
TMPDIR 9

X

X windows 21
X Windows 3, 18
xinfo 5, 6, 11
xrdb 3